Continuous After-the-fact Leakage-Resilient Key Exchange
(full version)

Janaka Alawatugoda[1] Colin Boyd[3] Douglas Stebila[1,2]

[1] *School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane, Australia*

[2] *Mathematical Sciences School, Queensland University of Technology, Brisbane, Australia*

janaka.alawatugoda@qut.edu.au, stebila@qut.edu.au

[3] *Department of Telematics, Norwegian University of Science and Technology, Trondheim, Norway*

colin.boyd@item.ntnu.no

Abstract

Security models for two-party authenticated key exchange (AKE) protocols have developed over time to provide security even when the adversary learns certain secret keys. In this work, we advance the modelling of AKE protocols by considering more granular, *continuous leakage* of long-term secrets of protocol participants: the adversary can adaptively request arbitrary leakage of long-term secrets even after the test session is activated, with limits on the amount of leakage per query but no bounds on the total leakage. We present a security model supporting continuous leakage even when the adversary learns certain ephemeral secrets or session keys, and give a generic construction of a two-pass leakage-resilient key exchange protocol that is secure in the model; our protocol achieves continuous, after-the-fact leakage resilience with not much more cost than a previous protocol with only bounded, non-after-the-fact leakage.

Keywords: leakage resilience, key exchange, continuous leakage, after-the-fact, security models

This is the full version of a paper published in the *Proceedings of the 19th Australasian Conference on Information Security and Privacy (ACISP 2014)* [2], in Wollongong, Australia, July 7-9, 2014, organized by the University of Wollongong, Australia.

Contents

1 Introduction

In order to capture leakage (side-channel) attacks, the notion of leakage resilience has been developed. Examples of information considered by leakage or side-channel attacks includes timing information [6, 10, 19], electromagnetic radiation [16], and power consumption information [23]. Leakage may reveal information about the secret parameters which have been used for computations in cryptosystems. To abstractly model leakage attacks, cryptographers have proposed the notion of *leakage-resilient* cryptography [1, 4, 9, 13, 14, 18, 17, 22], where the information that leaks is not fixed, but instead chosen adversarially. As authenticated key exchange is one of the most important cryptographic primitives in practice, it is important to construct key exchange protocols in a leakage-resilient manner.

Earlier key exchange security models, such as the Bellare–Rogaway [5], Canetti–Krawczyk [11], and extended Canetti–Krawczyk (eCK) [21] models, aim to capture security against an adversary who can fully compromise some, but not all secret keys. For example, in the eCK model, a session key should be secure even if the adversary has compromised either the long-term or ephemeral key at the client, and either the long-term or ephemeral key at the server, but not all of the values at one party. This is not a very granular form of leakage, and thus is not fully suitable for modelling side-channel attacks.

This motivates the development of leakage-resilient key exchange security models and protocols. Moriyama and Okamoto [25] and Alawatugoda, Stebila and Boyd [3] proposed key exchange security models to analyze security of leakage-resilient key exchange protocols, using a variant of the eCK model. There are two central limitations in the Moriyama–Okamoto model. First, the total amount of leakage allowed in the Moriyama–Okamoto model is bounded. Second, the adversary cannot obtain any leakage information after the "test" session is activated. The former restriction is troublesome because, in practice, ongoing executions of a protocol may reveal a small amount of leakage each time, and we would like to provide security against this "continuous" leakage. The latter restriction is problematic because we would like to provide security of one session, even if some leakage happens in subsequent sessions. Alawatugoda et al. [3] overcome the limitations of the Moriyama-Okamoto model by proposing a generic key exchange security model (ASB model), which can be instantiated using either continuous leakage model or bounded leakage, both instantiations allowing leakage after the "test" session is activated. Moreover, they proposed a generic construction of a protocol which can be proven secure in their generic model. However, concrete construction of their generic protocol with available cryptographic primitives can only be proven in the ASB bounded leakage model, because currently there exist no continuous leakage-resilient public-key cryptosystems. In this paper, we aim to propose a generic protocol which provides leakage resilience against continuous leakage, even after the "test" session is activated. In order to prove the security of our protocol, we use a slightly weakened variant of the ASB continuous leakage security model.

We now review few different approaches to modelling leakage. These leakage models generally allow the adversary to adaptively choose the leakage function that is evaluated against the long-term secret. The early leakage models generally did not allow leakage after a challenge had been issued, thus prevents the adversary from using subsequent calls to the leakage function to trivially solve the challenge. More recently, *after-the-fact leakage* schemes have been proposed to remove that restriction. We will review these schemes, then describe our contributions.

1.1 Leakage Models

In this section we review few leakage models, which have been widely used to define leakage-resilient security of cryptographic schemes.

Inspired by "cold boot" attacks, Akavia et al. [1] constructed a general framework to model memory attacks for public-key cryptosystems. With the knowledge of the public-key, the adversary can choose an efficiently computable arbitrary leakage function, f, and send it to the leakage oracle. The leakage oracle gives $f(sk)$ to the adversary where sk is the secret key. The only restriction here is that the sum of output length of all the leakage functions that an adversary can obtain is

bounded by some parameter λ which is smaller than the size of sk. This model is widely known as *bounded* leakage model.

In the work of Micali et al. [24], a general framework was introduced to model the leakage that occurs during computation with secret parameters. This framework relies on the assumption that *only computation leaks information* and that leakage only occurs from the secret memory portions which are actively involved in a computation. The adversary is allowed to obtain leakage from many computations. Therefore, the overall leakage amount is *unbounded* and in particular it can be larger than the size of the secret key.

Brakerski et al. [9] proposed a leakage model in which it is not assumed that the information is only leaked from the secret memory portions involved in computations. Instead it is assumed that leakage happens from the entire secret memory, but the amount of leakage is bounded per occurrence. In this model, number of leakage occurrences are allowed continuously. Therefore, the overall leakage amount is arbitrarily large. This model is widely known as *continuous* leakage model.

The above leakage models generally address the leakage which happens *before* the *challenge* is given to the adversary. In security experiments for public-key cryptosystems, the challenge is to distinguish the real plaintext corresponding to a particular ciphertext from a random plaintext, whereas in key exchange security models, the challenge is to identify the real session key of a chosen session from a random session key.

After-the-fact Leakage.

Leakage which happens after the *challenge* is given to the adversary can be considered as after-the-fact leakage. In leakage models for public-key cryptosystems, after-the-fact leakage is the leakage which happens after the challenge ciphertext is given whereas in leakage security models for key exchange protocols, after-the-fact leakage is the leakage which happens after the test session is activated.

For leakage-resilient public-key encryption there are three properties which may be important differentiators for the different models. One is whether the model allows access to decryption of chosen ciphertexts before (CCA1) or after (CCA2) the challenge is known. The second is whether the leakage allowed to the adversary is *continuous* or *bounded*. The third is whether the leakage is allowed only before the challenge ciphertext is known or also after the fact.

In earlier models, such as that of Naor et al. [26], it was expected that although the adversary is given access to the decryption oracle (CCA2), the adversary cannot be allowed to obtain leakage after the challenge ciphertext is given. This is because the adversary can encode the decryption algorithm and challenge ciphertext with the leakage function and by revealing a few bits of the decrypted value of the challenge ciphertext trivially win the challenge. Subsequently, Halevi et al. [15] introduced chosen plaintext after-the-fact leakage-resilient security on public-key cryptosystems. In their security experiment, the adversary is not allowed to access the decryption oracle. Further, the total leakage amount is also bounded.

Dziembowski et al. [12] defined an adaptively chosen ciphertext attack (CCA2) security experiment in which the adversary is allowed to obtain leakage information even after the challenge ciphertext is given. Their security experiment defines adaptively chosen ciphertext after-the-fact leakage (CCLA2) which can be considered as the strongest security notion of public-key cryptosystems; it allows the adversary adaptive access to the decryption oracle and leakage information even after the challenge ciphertext is given. Furthermore, they allow continuous leakage so the total leakage amount is unbounded. This is achieved by keeping the secret key in a split state, an idea earlier introduced by Kiltz et al. [18], using the reasonable assumption that leakage occurs only when computation takes place, leakage is only bounded per invocation of the secret key while the state is updated after each invocation.

Recall that in key exchange security models, the challenge to the adversary is to distinguish the real session key of a chosen session from a random session key. In the Moriyama–Okamoto [25] key exchange security model, the adversary is not allowed to obtain leakage after the test session

Security model	SessionKey	EphemeralKey	Corrupt	Combinations	Leakage resilience
eCK [21]	Yes	Yes	Yes	4/4	None
MO [25]	Yes	Yes	Yes	4/4	Bounded, not after-the-fact
ASB [3]	Yes	Yes	Yes	4/4	Bounded/Continuous, after-the-fact
CAFL (this paper)	Yes	Yes	Yes	2/4	Continuous, after-the-fact

Table 1: Key exchange security models with reveal queries and leakage allowed

Protocol	Initiator cost	Responder cost	Security model	Proof model
NAXOS [21]	4 **Exp**	4 **Exp**	eCK	Random oracle
MO [25]	8 **Exp**	8 **Exp**	MO	Standard
ASB [3]	12 **Exp**	12 **Exp**	ASB (Bounded)	Standard
π instantiation	10 **Exp**	10 **Exp**	CAFL	Standard

Table 2: Security and efficiency comparison of key exchange protocols

is activated, whereas in the ASB model, the adversary is allowed to obtain leakage even after the test session is activated.

In the literature there are no key exchange protocols available that are secure against continuous leakage after the test session is activated. Alawatugoda et al. [3] proposed a generic construction of a key exchange protocol which provides security against leakage after the test session is activated, but when instantiated with available cryptographic primitives it does not provide continuous leakage resilience.

1.2 Our Contribution

Alawatugoda et al. [3] mentioned that constructing a continuous after-the-fact leakage-resilient key exchange protocol in the ASB continuous leakage model is an open problem. In this paper, we aim to construct a continuous after-the-fact leakage-resilient key exchange protocol using existing leakage-resilient cryptographic primitives. In order to prove the security of our protocol, we use a *weaker* variant of the generic ASB model's continuous leakage instantiation. The meaning of "weaker" is defined by means of the freshness condition. While weakening a model is generally undesirable, introducing the restrictions allow us to actually achieve the security definition, whereas no instantiation of the ASB continuous leakage-resilient key exchange protocol is known. Thus, we begin by presenting the **continuous after-the-fact leakage model** (CAFL model).

Table 1 summarizes the adversarial powers of CAFL model in comparison with the adversarial powers of CK model [11], eCK model [21] and Moriyama–Okamoto (MO) model [25] and the generic Alawatugoda–Stebila–Boyd (ASB) [3] model. There are four `Corrupt`–`EphemeralKeyReveal` query combinations which do not trivially expose the session key. In the column "Combinations" of Table 1, we mention how many of them are allowed in the corresponding security model. We discuss more about query combinations in detail in Section 3.3.

We then construct a generic protocol π which can be proven secure in this model; the protocol is a "key agreement"-style protocol, and it relies on a public-key cryptosystem that is secure against adaptively chosen ciphertext attacks with after-the-fact leakage-resilience (abbreviated as CCLA2). In Table 2, we compare an instantiation of the proposed generic protocol π, with the NAXOS protocol [21], the Moriyama-Okamoto (MO) protocol [25] and the generic ASB protocol instantiation, by means of computation cost, security model and the proof model. The protocol π is instantiated using the CCLA2-secure public-key cryptosystem of Dziembowski et al. [12].

Table 2 shows that the instantiation of protocol π provides significant leakage resilience properties for practically achievable computation costs, and thus π is a viable framework for construction of CAFL-secure protocols. The generic protocol π can be instantiated with any CCLA2-secure public-key cryptosystem. Our proof shows that protocol π can achieve the same leakage tolerance

as the underlying public-key cryptosystem tolerates. Moreover, protocol π can be instantiated with smaller computational cost by using *cost effective* CCLA2-secure public-key encryption schemes.

2 Background

In this section we review the formal definitions of the tools we will use to construct our protocol.

2.1 CCLA2-Secure Public-Key Cryptosystems

Dziembowski et al. [12] constructed an adaptively chosen ciphertext after-the-fact leakage-resilient public-key cryptosystem which is secure against continuous leakage.

Definition 2.1 (Security Against Adaptively Chosen Ciphertext After-the-fact Leakage Attacks (CCLA2))**.** Let $k \in \mathbb{N}$ be the security parameter. A public-key cryptosystem $\mathrm{PKE} = (\mathrm{KG}, \mathrm{Enc}, \mathrm{Dec})$ is λ-CCLA2 secure if for any probabilistic polynomial time adversary $\mathcal{D}$, the advantage of winning the following distinguishing game is negligible.

1. $(sk, pk) \leftarrow \mathrm{KG}(1^k)$.
2. $(m_0, m_1, state) \leftarrow \mathcal{D}^{\mathrm{Leak}(.),\mathrm{Dec}(sk,.)}(pk)$ such that $|m_0| = |m_1|$
3. $b \leftarrow \{0, 1\}$
4. $C \leftarrow \mathrm{Enc}(pk, m_b)$
5. $b' \leftarrow \mathcal{D}^{\mathrm{Leak}(.),\mathrm{Dec}_{\neq C}(sk,.)}(C, state)$
6. Output b'. $\mathcal{D}$ wins if $b' = b$.

Decryption Oracle
$\mathrm{Dec}(sk, c) \rightarrow (sk', m)$ where m is the corresponding plaintext of the ciphertext c.
Update the secret state sk to sk'.

Leakage Oracle
For any adversary chosen efficiently computable leakage function f, $\mathrm{Leak}(f) \rightarrow f(sk)$ whenever $|f(sk)| \leq \lambda$. The **Leakage Oracle** is called whenever the **Decryption Oracle** is called.

In the Dziembowski et al. [12] public-key cryptosystem, the secret key $sk = (x_1, x_2) \in \mathbb{Z}_p^2$ is split into two parts ℓ_{sk}, r_{sk} such that $\ell_{sk} \leftarrow \mathbb{Z}^n$ at random and $r_{sk} \leftarrow \mathbb{Z}^{n \times 2}$ holding $\ell_{sk} \cdot r_{sk} = sk$, where n is the statistical security parameter. They proved their public-key cryptosystem is CCLA2 secure for $\lambda = 0.15 \cdot n \cdot \log p - 1$. So if we consider $n = 80$ and $\log p$ to be 1024, we can allow $\lambda = 12276$ bits of leakage. Considering only the most expensive computations, the computation cost of Enc and Dec is $5\mathbf{Exp}$ where **Exp** is the computational cost of an exponentiation.

2.2 Key Derivation Functions

We review the definitions of key derivation functions proposed by Krawczyk [20]. Secure and efficient key derivation functions are available in the literature, for example based on HMAC [20].

Definition 2.2 (Key Derivation Function)**.** A key derivation function KDF is an efficient algorithm that accepts as input four arguments: a value σ sampled from a source of keying material Σ, a length value k and two additional arguments, a salt value r defined over a set of possible salt values and a context variable c, both of which are optional i.e., can be set to a null. The KDF output is a string of k bits.

Definition 2.3 (Source of Key Material)**.** A source of keying material Σ is a two-valued (σ, κ) probability distribution generated by an efficient probabilistic algorithm, where σ is the secret source key material to be input to the KDF and κ is some public knowledge about σ or its distribution.

Definition 2.4 (Security of key derivation function w.r.t a source of key material)**.** A key derivation function KDF is said to be secure with respect to a source of key material Σ if no feasible attacker $\mathcal{B}$ can win the following distinguishing game with probability significantly better than 1/2:

1. $(\sigma, \kappa) \leftarrow \Sigma$. (Both the probability distribution as well as the generating algorithm have been referred by Σ)
2. A salt value r is chosen at random from the set of possible salt values defined by KDF (r may be set to a constant or a null value if so defined by KDF).
3. The attacker $\mathcal{B}$ is provided with κ and r.
4. $\mathcal{B}$ chooses arbitrary value k and c.
5. A bit $b \leftarrow \{0, 1\}$ is chosen at random. If $b = 0$, attacker $\mathcal{B}$ is provided with the output of $\text{KDF}(\sigma, r, k, c)$ else $\mathcal{B}$ is given a random string of k bits.
6. $\mathcal{B}$ outputs a bit $b' \leftarrow \{0, 1\}$. $\mathcal{B}$ wins if $b' = b$.

2.3 Decision Diffie-Hellman Problem

The decision Diffie-Hellman (DDH) problem is a computational hardness assumption based on discrete logarithms in a cyclic group [7]. Consider a cyclic group $\mathbb{G}$ of order q, with a generator g. For $a, b, c \in \mathbb{Z}_p$, the DDH problem is to distinguish the triple (g^a, g^b, g^{ab}) from the triple (g^a, g^b, g^c).

3 Continuous After-the-fact Leakage Model

A *key agreement* protocol is an interactive protocol executed between two parties to establish a shared secret key. In this section we introduce the continuous after-the-fact leakage model, (CAFL model), for key exchange. In the CAFL model, the adversary is allowed to adaptively obtain partial leakage on the long-term secret keys even after the test session is activated, as well as reveal session keys, long-term keys, and ephemeral keys.

3.1 Protocol Execution

Parties and Long-term Keys.

Let $\mathcal{U} = \{U_1, \ldots, U_{N_P}\}$ be a set of N_P parties. Each party U_i where $i \in [1, N_P]$ has a pair of long-term public and secret keys, (pk_{U_i}, sk_{U_i}).

Sessions.

Each party may run multiple instances of the protocol concurrently or sequentially; we use the term *principal* to refer a party involved in a protocol instance, and the term *session* to identify a protocol instance at a principal. The notation $\Pi^s_{U,V}$ represents the s^{th} session at the owner principal U, with intended partner principal V. The principal which sends the first protocol message of a session is the *initiator* of the session, and the principal which responds to the first protocol message is the *responder* of the session. A session $\Pi^s_{U,V}$ enters an *accepted* state when it computes a session key. Note that a session may terminate without ever entering into the accepted state. The information of whether a session has terminated with or without acceptance is public.

Adversary Interaction.

The adversary (a probabilistic algorithm) controls all interaction and communication between parties. In particular, the adversary initiates sessions at parties and delivers protocol messages; it can create, change, delete, or reorder messages. The adversary can also compromise certain short-term and long-term secrets. Notably, whenever the party performs an operation using its long-term key, the adversary obtains some leakage information about the long-term key.

The following query allows the adversary $\mathcal{A}$ to run the protocol, modelling normal communication.

- `Send`$(U, V, s, m, \mathbf{f})$ query: The oracle $\Pi^s_{U,V}$, computes the next protocol message according to the protocol specification on receipt of m, and sends it to the adversary $\mathcal{A}$, along with the leakage $\mathbf{f}(sk_U)$ as described in Section 3.2. $\mathcal{A}$ can also use this query to activate a new protocol instance as an initiator with blank m and $\mathbf{f}$.

The following queries allow the adversary $\mathcal{A}$ to compromise certain session specific ephemeral secrets and long-term secrets from the protocol principals.

- `SessionKeyReveal`(U, V, s) query: $\mathcal{A}$ is given the session key of the oracle $\Pi^s_{U,V}$, if the oracle $\Pi^s_{U,V}$ is in the accepted state.
- `EphemeralKeyReveal`(U, V, s) query: $\mathcal{A}$ is given the ephemeral keys of the oracle $\Pi^s_{U,V}$.
- `Corrupt`(U) query: $\mathcal{A}$ is given the long-term secrets of the principal U. This query does not reveal any session keys or ephemeral keys to $\mathcal{A}$.

3.2 Modelling Leakage

In this key exchange security model we consider *continuous leakage of the long-term secret keys* of protocol principals, because long-term secret keys are not one-time secrets, but they last for multiple protocol sessions. Leakage of long-term secret key from one session affects to the security of another session which uses the same long-term secret key. Considering side-channel attacks which can be mounted against key exchange protocols, the most realistic way to obtain the leakage information of long-term secret keys is from the protocol computations which use long-term secret keys. Hence, following the premise "only computation leaks information" [24], we have modeled the leakage to occur where computation takes place using secret keys. By issuing a `Send` query, the adversary will get a protocol message which is computed according to the normal protocol computations. Therefore, the instance of a `Send` query would be the appropriate instance to address the leakage occurs due to a computation which uses a long-term secret key. Thus, sending an adversary-chosen leakage function, $\mathbf{f}$, with the `Send` query would reflect the premise "only computation leaks information".

Further, we assume that the amount of leakage of a secret key is bounded by a leakage parameter λ, per computation. The adversary is allowed to obtain leakage from many computations continuously. Hence, the overall leakage amount is unbounded.

Remark 1 (`Corrupt` query vs Leakage queries). By issuing a `Corrupt` query, the adversary gets the party's entire long-term secret key. Separately, by issuing leakage queries (using leakage function $\mathbf{f}$ embedded with the `Send` query) the adversary gets λ-bounded amount of leakage information about the long-term secret key. It may seem paradoxical to consider `Corrupt` and Leakage queries at the same time. But there are good reasons to consider both.

- A *non-leakage* version of CAFL model (`Send` query without $\mathbf{f}$) addresses KCI attacks, because the adversary is allowed to corrupt the owner of the test session before the activation of the test session. In the CAFL model, we allow the adversary to obtain leakage from the partner of the test session, in addition to allowing the adversary to corrupt the owner of the test session.
- A *non-leakage* version of CAFL model (`Send` query without $\mathbf{f}$) addresses partial weak forward secrecy, because the adversary is allowed to corrupt either of the protocol principals, but not both, after the test session is activated. In the CAFL model, we allow the adversary to obtain leakage from the uncorrupted principal, in addition to allowing the adversary to corrupt one of the protocol principals.

Hence, the CAFL model allows the adversary to obtain more information than a *non-leakage* version of CAFL model.

3.3 Defining Security

In this section we give formal definitions for partner sessions, freshness of a session and security in the CAFL model.

Definition 3.1 (Partner sessions in CAFL model)**.** Two oracles $\Pi^{s}_{U,V}$ and $\Pi^{s'}_{U',V'}$ are said to be partners if:

1. $\Pi^{s}_{U,V}$ and $\Pi^{s'}_{U',V'}$ have computed session keys and
2. Sent messages from $\Pi^{s}_{U,V}$ = Received messages to $\Pi^{s'}_{U',V'}$ and
3. Sent messages from $\Pi^{s'}_{U',V'}$ = Received messages to $\Pi^{s}_{U,V}$ and
4. $U' = V$ and $V' = U$ and
5. If U is the initiator then V is the responder, or vise versa.

A protocol is said to be *correct* if two partner oracles compute identical session keys in the presence of a passive adversary. Once the oracle $\Pi^{s}_{U,V}$ has accepted a session key, asking the following query the adversary $\mathcal{A}$ attempt to distinguish it from a random session key. The `Test` query is used to formalize the notion of the semantic security of a key exchange protocol.

- `Test`(U, V, s) query: When $\mathcal{A}$ asks the `Test` query, the oracle $\Pi^{s}_{U,V}$ first chooses a random bit $b \leftarrow \{0, 1\}$ and if $b = 1$ then the actual session key is returned to $\mathcal{A}$, otherwise a random string chosen from the same session key space is returned to $\mathcal{A}$. This query is only allowed to be asked once across all sessions.

We now define what it means for a session to be λ-CAFL-*fresh* in the CAFL model.

Definition 3.2 (λ-CAFL-freshness)**.** Let λ be the leakage bound per occurrence. An oracle $\Pi^{s}_{U,V}$ is said to be λ-CAFL-*fresh* if and only if:

1. The oracle $\Pi^{s}_{U,V}$ or its partner, $\Pi^{s'}_{V,U}$ (if it exists) has not been asked a `SessionKeyReveal`.
2. If the partner $\Pi^{s'}_{V,U}$ exists, none of the following combinations have been asked:
   (a) `Corrupt`(U) and `Corrupt`(V).
   (b) `Corrupt`(U) and `EphemeralKeyReveal`(U, V, s).
   (c) `Corrupt`(V) and `EphemeralKeyReveal`(V, U, s').
   (d) `EphemeralKeyReveal`(U, V, s) and `EphemeralKeyReveal`(V, U, s').
3. If the partner $\Pi^{s'}_{V,U}$ does not exist, none of the following combinations have been asked:
   (a) `Corrupt`(V).
   (b) `Corrupt`(U) and `EphemeralKeyReveal`(U, V, s).
4. For each `Send`$(\cdot, U, \cdot, \cdot, \mathbf{f})$ query, the output of $\mathbf{f}$ is at most λ bits.
5. For each `Send`$(\cdot, V, \cdot, \cdot, \mathbf{f})$ query, the output of $\mathbf{f}$ is at most λ bits.

When the adversary asks `EphemeralKeyReveal` and `Corrupt` queries, there are two `Corrupt`–`EphemeralKeyReveal` query combinations which trivially expose the session key of an oracle.

1. `Corrupt`(U) and `EphemeralKeyReveal`(U, V, s).
2. `Corrupt`(V) and `EphemeralKeyReveal`(V, U, s').

As in the other models we have compared with [21, 25, 3] we do not allow above combinations in the freshness condition, as they trivially expose the session key of oracles $\Pi^{s}_{U,V}$ and $\Pi^{s'}_{V,U}$. Differently, there are four `Corrupt`–`EphemeralKeyReveal` query combinations which do not trivially expose the session key an oracle.

1. `Corrupt`(U) and `Corrupt`(V).
2. `Corrupt`(U) and `EphemeralKeyReveal`(V, U, s).
3. `Corrupt`(V) and `EphemeralKeyReveal`(U, V, s').
4. `EphemeralKeyReveal`(V, U, s) and `EphemeralKeyReveal`(U, V, s').

All the models we consider [21, 25, 3] allow above combinations in the freshness condition, whereas our CAFL model does not allow the query combinations 1 and 4 in the freshness condition. In that sense the CAFL model is weaker than the ASB continuous leakage model.

Security of a key exchange protocol in the CAFL model is defined using the following security game, which is played by a probabilistic polynomial time adversary $\mathcal{A}$ against the protocol challenger.

- **Stage 1:** $\mathcal{A}$ may ask any of `Send`, `SessionKeyReveal`, `EphemeralKeyReveal` and `Corrupt` queries to any oracle at will.
- **Stage 2:** $\mathcal{A}$ chooses a λ-CAFL-fresh oracle and asks a `Test` query.
- **Stage 3:** $\mathcal{A}$ may continue asking `Send`, `SessionKeyReveal`, `EphemeralKeyReveal` and `Corrupt` queries. $\mathcal{A}$ may not ask a query that violates the λ-CAFL-freshness of the test session.
- **Stage 4:** Eventually, $\mathcal{A}$ outputs the bit $b' \leftarrow \{0, 1\}$ which is its guess of the value b on the test session. $\mathcal{A}$ wins if $b' = b$.

$Succ(\mathcal{A})$ is the event that $\mathcal{A}$ wins the above security game. The definition of security follows.

Definition 3.3 (λ-CAFL-security)**.** A protocol π is said to be λ-CAFL-secure if there is no probabilistic polynomial time algorithm $\mathcal{A}$ that can win the above game with non-negligible advantage. The advantage of an adversary $\mathcal{A}$ is defined as $Adv^{\mathrm{CAFL}}_{\pi}(\mathcal{A}) = |2\Pr(\mathrm{Succ}_{\mathcal{A}}) - 1|$.

3.4 Practical Interpretation of Security of CAFL Model.

We review the relationship between the CAFL model and real world attack scenarios.

- **Active adversarial capabilities:** `Send` queries address the powers of an active adversary who can control the message flow over the network.
- **Side-channel attacks:** Leakage functions are embedded with the `Send` query. Thus, a wide variety of side-channel attacks based on **continuous leakage of long-term secrets** are addressed, assuming that the leakage happens when computations take place in principals.
- **Cold-boot attacks:** `Corrupt` queries address situations which reveal the long-term secret keys of protocol principals like in cold-boot attacks.
- **Malware attacks:** `EphemeralKeyReveal` queries cover the malware attacks which steal stored ephemeral keys, given that the long-term keys may be securely stored separately from the ephemeral keys in places such as smart cards or hardware security modules. Separately, `Corrupt` queries address malware attacks which steal the long-term secret keys of protocol principals.

- **Weak random number generators:** After knowing a previous set of randomly generated ephemeral values the adversary may be able to identify the statistical pattern of the random number generator and hence correctly guess the next value with a high probability. `EphemeralKeyReveal` query addresses situations where the adversary can get the ephemeral secrets.
- **Known key attacks:** `SessionKeyReveal` query covers the attacks which can be mounted by knowing past session keys.
- **Key compromise impersonation attacks:** λ-CAFL-freshness allows the adversary to corrupt the owner of the test session before the activation of the test session. Hence, the CAFL model security protects against the key compromise impersonation attacks.
- **Partial weak forward secrecy:** λ-CAFL-freshness allows the adversary to corrupt either of the protocol principals, but not both, after the test session is activated. Hence, the CAFL model addresses partial weak forward secrecy.

Although our model is a weaker variant of the ASB continuous leakage model, it addresses all the attack scenarios which are addressed by the ASB model, except *weak forward secrecy*. Instead, our model addresses *partial weak forward secrecy*. Hence, our model is very similar to the generic ASB model and interprets most of real world attack scenarios.

4 Protocol π

In Table 3 we show the generic construction of protocol π. Enc and Dec are the encryption and decryption algorithms of the underlying adaptively chosen ciphertext after-the-fact leakage (CCLA2) secure public-key cryptosystem, PKE. KDF is a secure key derivation function which generates the session key of length k. The protocol π is a key agreement protocol, in which each of the principals randomly chooses its ephemeral secret key, encrypts it with the public-key of the intended partner principal using the encryption algorithm Enc, and sends the encrypted message to the intended partner principal. After exchanging the ephemeral secrets both principals compute the session key with ephemeral secrets and identities of the two principals, using the key derivation function KDF. We underlined the computations which could leak information about secret keys.

A (Initiator)		B (Responder)
	Initial Setup	
$sk_A, pk_A \leftarrow \mathrm{KG}(1^k)$		$sk_B, pk_B \leftarrow \mathrm{KG}(1^k)$
	Protocol Execution	
$r_A \leftarrow \{0,1\}^k$		$r_B \leftarrow \{0,1\}^k$
$C_A \leftarrow \mathrm{Enc}(pk_B, r_A)$	$\xrightarrow{A, C_A}$	$(sk'_B, r_A) \leftarrow \underline{\mathrm{Dec}}(sk_B, C_A)$
		$sk_B \leftarrow sk'_B$
$(sk'_A, r_B) \leftarrow \underline{\mathrm{Dec}}(sk_A, C_B)$	$\xleftarrow{B, C_B}$	$C_B \leftarrow \mathrm{Enc}(pk_A, r_B)$
$sk_A \leftarrow sk'_A$		
$K_{AB} \leftarrow \mathrm{KDF}(A, B, r_A, r_B)$		$K_{AB} \leftarrow \mathrm{KDF}(A, B, r_A, r_B)$
	K_{AB} is the session key	

Table 3: Protocol π. Underline denotes operations to which leakage functions apply.

Theorem 4.1. *The protocol π is λ-CAFL-secure, whenever the underlying public-key cryptosystem* PKE *is* CCLA2 *secure and the key derivation function* KDF *is secure with respect to a uniformly random key material.*

In order to formally prove the CAFL-security of the protocol π we use the game hopping technique [27]; define a sequence of games and relate the adversary's advantage of distinguishing each game from the previous game to the advantage of breaking one of the underlying cryptographic primitive. The proof structure is similar to Boyd et al. [8]. The security proof of Theorem 4.1 is available in Appendix A.

5 Conclusion and Future Work

We have proposed a key exchange protocol and a security model that improves the amount and type of secret leakage. Our protocol is a generic key exchange protocol, that relies on a continuous after-the-fact leakage-resilient public-key encryption scheme. Using such schemes from the literature, our protocol can be instantiated without much more cost than previous schemes which tolerate only bounded leakage. Our security model allows the adversary to fully compromise a variety of long-term and short-term ephemeral values, as well as obtain partial, adaptive, continuous, after-the-fact leakage of long-term secret keys. Our model captures a wide variety of practical attack scenarios, including cold boot, key compromise impersonation, and side channel attacks.

The challenge is to achieve a secure protocol in the ASB continuous leakage model. The ASB continuous leakage model is a continuous leakage variant of the eCK model. There are two main techniques for constructing (non-leakage-resilient) eCK-secure protocols: use of the so-called "NAXOS trick", in which the long-term and ephemeral secret keys are hashed together to derive the ephemeral Diffie–Hellman exponent, and MQV-style protocols, which algebraically combine ephemeral and static Diffie–Hellman computations. Since the NAXOS trick involves a calculation based on the secret key, adapting such a protocol requires the use of a continuous leakage-resilient NAXOS trick. By using pair-generation-indistinguishable continuous-after-the-fact-leakage-resilient public-key cryptosystems, it would be possible to obtain a continuous leakage-resilient NAXOS trick as shown in Alawatugoda et al. [3]. A leakage-resilient protocol based on MQV-style computations is also an interesting open question.

References

[1] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *Theory of Cryptology Conference*, pages 474–495, 2009.

[2] J. Alawatugoda, C. Boyd, and D. Stebila. Continuous after-the-fact leakage-resilient key exchange. In *ACISP*, 2014.

[3] J. Alawatugoda, D. Stebila, and C. Boyd. Modelling after-the-fact leakage for key exchange (full version). *IACR Cryptology ePrint Archive*, Report 2014/131, 2014.

[4] J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.

[5] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *CRYPTO*, pages 232–249, 1993.

[6] D. J. Bernstein. Cache-timing attacks on AES. Technical report, 2005. http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

[7] D. Boneh. The decision Diffie-Hellman problem. In *Algorithmic Number Theory Symposium*, pages 48–63, 1998.

[8] C. Boyd, Y. Cliff, J. M. G. Nieto, and K. G. Paterson. One-round key exchange in the standard model. *International Journal of Advanced Computer Technology*, pages 181–199, 2009.

[9] Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. *IACR Cryptology ePrint Archive*, Report 2010/278, 2010.

[10] D. Brumley and D. Boneh. Remote timing attacks are practical. In *USENIX Security Symposium*, pages 1–14, 2003.

[11] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT*, pages 453–474, 2001.

[12] S. Dziembowski and S. Faust. Leakage-resilient cryptography from the inner-product extractor. In *ASIACRYPT*, pages 702–721, 2011.

[13] S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *IEEE Symposium on Foundations of Computer Science*, pages 293–302, 2008.

[14] S. Faust, E. Kiltz, K. Pietrzak, and G. N. Rothblum. Leakage-resilient signatures. *IACR Cryptology ePrint Archive*, Report 2009/282, 2009.

[15] S. Halevi and H. Lin. After-the-fact leakage in public-key encryption. In *Theory of Cryptology Conference*, pages 107–124, 2011.

[16] M. Hutter, S. Mangard, and M. Feldhofer. Power and EM attacks on passive 13.56MHz RFID devices. In *CHES*, pages 320–333, 2007.

[17] J. Katz and V. Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.

[18] E. Kiltz and K. Pietrzak. Leakage resilient elgamal encryption. In *ASIACRYPT*, pages 595–612, 2010.

[19] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO*, pages 104–113, 1996.

[20] H. Krawczyk. On extract-then-expand key derivation functions and an HMAC-based KDF. http://webee.technion.ac.il/ hugo/kdf/kdf.pdf, 2008.

[21] B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *ProvSec*, pages 1–16, 2007.

[22] T. Malkin, I. Teranishi, Y. Vahlis, and M. Yung. Signatures resilient to continual leakage on memory and computation. In *Theory of Cryptology Conference*, pages 89–106, 2011.

[23] T. Messerges, E. Dabbish, and R. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*, pages 541–552, 2002.

[24] S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In *Theory of Cryptology Conference*, pages 278–296, 2004.

[25] D. Moriyama and T. Okamoto. Leakage resilient eCK-secure key exchange protocol without random oracles. In *ASIACCS*, pages 441–447, 2011.

[26] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35. 2009.

[27] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, Report 2004/332, 2004.

A Security Proof

Proof. Assume that the adversary $\mathcal{A}$ can win the challenge against the protocol π challenger with non negligible advantage $Adv_{\pi}^{\mathrm{CAFL}}(\mathcal{A})$. We split the proof into two cases: when the owner of the test session is corrupted, and when it is not.

Case 1: The owner of the test session is corrupted

In this case we consider the situation that $\mathcal{A}$ corrupts the owner of the test session but not the partner.

Game 1:

This game is the original game. When the `Test` query is asked, the Game 1 challenger chooses a random bit $b \leftarrow \{0,1\}$. If $b = 1$, the real session key is given to $\mathcal{A}$, otherwise a random value chosen from the same session key space is given.

Game 2:

Same as Game 1 with the following exception: before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \leftarrow \{U_1, \ldots, U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \leftarrow \{1, \ldots, N_s\}$ are chosen, where N_P is the number of protocol principals and N_S is the number of sessions on a principal. The oracle $\Pi^{s^*}_{U^*,V^*}$ is chosen as the *target session* and the oracle $\Pi^{t^*}_{V^*,U^*}$ is chosen as the *partner* to the target session. If the test session is *not* the oracle $\Pi^{s^*}_{U^*,V^*}$ or the partner to the oracle is not $\Pi^{t^*}_{V^*,U^*}$, the Game 2 challenger aborts the game.

Game 3:

Same as Game 2 with the following exception: the Game 3 challenger chooses a random value $r' \leftarrow \{0,1\}^k$.

- If the test session is on the initiator, the challenger computes $K_{U^*V^*} \leftarrow \mathrm{KDF}(U^*, V^*, r', r_{V^*})$.
- If the test session is on the responder, the challenger computes $K_{U^*V^*} \leftarrow \mathrm{KDF}(V^*, U^*, r_{V^*}, r')$.

Game 4:

Same as Game 3 with the following exception: the Game 4 challenger randomly chooses $K \leftarrow \{0,1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the `Test` query.

Differences between games

In this section the adversary's advantage of distinguishing each game from the previous game is investigated. $Succ_{\text{Game x}}(\mathcal{A})$ denotes the event that the adversary $\mathcal{A}$ wins Game x, $Adv_{\text{Game x}}(\mathcal{A})$ denotes the advantage of the adversary $\mathcal{A}$ of winning Game x.

Game 1

is the original game. Hence,

$$Adv_{\text{Game 1}}(\mathcal{A}) = Adv_{\pi}^{\mathrm{CAFL}}(\mathcal{A}). \tag{1}$$

Game 1 and Game 2.

The probability of Game 2 to be halted due to incorrect choice of the test session is $1 - \frac{1}{N_P^2 N_s^2}$. Unless the incorrect choice happens, Game 2 is identical to Game 1. Hence,

$$Adv_{\text{Game 2}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} Adv_{\text{Game 1}}(\mathcal{A}). \tag{2}$$

Game 2 and Game 3.

We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$. If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{D}$ can be used against the CCLA2 challenger of underlying public-key cryptosystem, PKE. The algorithm $\mathcal{D}$ uses the public-key of the CCLA2 challenger as the public-key of the protocol principal V^* and generates public/secret key pairs for all other protocol principals. $\mathcal{D}$ runs a copy of $\mathcal{A}$ and interacts with $\mathcal{A}$, such that it is interacting with either Game 2 or Game 3. $\mathcal{D}$ picks two random strings, $r'_0, r'_1 \leftarrow \{0,1\}^k$ and passes them to the CCLA2 challenger. From the CCLA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C such that $C \leftarrow \text{Enc}(pk_{V^*}, r')$ where $r' = r'_0$ or $r' = r'_1$. The following describes the procedure of answering queries.

- $\texttt{Send}(U, V, s, m, \mathbf{f})$ query: If $U = U^*$, $V = V^*$ and $s = t^*$, then $\mathcal{D}$ sends the ciphertext C to $\mathcal{A}$ with the leakage $\mathbf{f}(sk_{U^*})$. Else if $U \neq V^*$, then $\mathcal{D}$ randomly picks a value $r_U \leftarrow \{0,1\}^k$, encrypts it with the public-key of V and sends it to $\mathcal{A}$ with the leakage $\mathbf{f}(sk_U)$.

  If $U = V^*$, then $\mathcal{D}$ sends the ciphertexts which comes to the principal V^*, to the PKE challenger with the leakage function $\mathbf{f}$. The PKE challenger decrypts the ciphertext and sends the corresponding plaintext to $\mathcal{D}$ with the leakage, $\mathbf{f}(sk_{V^*})$. $\mathcal{D}$ randomly picks a value $r_{V^*} \leftarrow \{0,1\}^k$, encrypts it with the public-key of V and sends it to $\mathcal{A}$ with the leakage $\mathbf{f}(sk_{V^*})$.
- $\texttt{SessionKeyReveal}(U, V, s)$ query: If $U = V^*$ or $V = V^*$, $\mathcal{D}$ uses the PKE challenger to decrypt the ciphertext which is encrypted by the public-key of V^* and compute the session key by $\text{KDF}(V^*, V, r_{V^*}, r_V)$ if V^* is the initiator or $\text{KDF}(U, V^*, r_U, r_{V^*})$ if V^* is the responder. If $U \neq V^*$ and $V \neq V^*$, $\mathcal{D}$ can decrypt both ciphertexts from the both principals by its own, hence recover r_U and r_V and compute the session key by $\text{KDF}(U, V, r_U, r_V)$.
- $\texttt{EphemeralKeyReveal}(U, V, s)$ query: For all legitimate $\texttt{EphemeralKeyReveal}$ queries $\mathcal{D}$ will answer with the ephemeral-key.
- $\texttt{Corrupt}(U)$ query: Except for V^*, algorithm $\mathcal{D}$ can answer all other $\texttt{Corrupt}$ queries. In this case we consider the situation in which the adversary not allowed to corrupt the partner principal of the target session.
- $\texttt{Test}(U, V, s)$ query: To compute the answer to the $\texttt{Test}(U^*, V^*, s^*)$ query, the algorithm $\mathcal{D}$ uses r'_1 as the decryption of the ciphertext C and computes $\text{KDF}(U^*, V^*, r'_1, r_{V^*})$, if U^* is the initiator or computes $\text{KDF}(V^*, U^*, r_{V^*}, r'_1)$, if V^* is the initiator.

If r'_1 is the decryption of C coming *from* the owner of the test session, U^*, the simulation constructed by $\mathcal{D}$ is identical to Game 2 whereas if r'_0 is the decryption of C, the simulation constructed by $\mathcal{D}$ is identical to Game 3. If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{D}$ can distinguish whether $C \leftarrow \text{Enc}(pk_{V^*}, r'_0)$ or $C \leftarrow \text{Enc}(pk_{V^*}, r'_1)$.

The algorithm $\mathcal{D}$ plays the CCLA2 game against the public-key cryptosystem PKE according to the Definition 2.1 since $\mathcal{D}$ does not ask the decryption of the challenge ciphertext C. Hence,

$$|Adv_{\text{Game 2}}(\mathcal{A}) - Adv_{\text{Game 3}}(\mathcal{A})| \leq Adv^{\text{PKE}}(\mathcal{D}). \tag{3}$$

Game 3 and Game 4.

We introduce an algorithm $\mathcal{B}$ which is constructed using the adversary $\mathcal{A}$. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{B}$ can be used to distinguish whether the value K is computed using KDF or randomly chosen. $\mathcal{B}$ receives K from the KDF challenger, such that K is computed using the KDF or randomly chosen from the session key space. If K is computed using the KDF, the simulation constructed by $\mathcal{B}$ is identical to Game 3 whereas if K is randomly chosen, the simulation constructed by $\mathcal{B}$ is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{B}$ can distinguish whether the value K is computed using KDF or randomly chosen. Hence,

$$|Adv_{\text{Game 3}}(\mathcal{A}) - Adv_{\text{Game 4}}(\mathcal{A})| \leq Adv^{\text{KDF}}(\mathcal{B}). \tag{4}$$

Semantic security of the session key in Game 4.

Since the session key K of $\Pi^{s^*}_{U^*,V^*}$ is chosen randomly and independently from all other values, $\mathcal{A}$ does not have any advantage in Game 4. Hence,

$$Adv_{\text{Game 4}}(\mathcal{A}) = 0. \tag{5}$$

Using equations (1)–(5) we find,

$$Adv^{\text{CAFL}}_{\pi}(\mathcal{A}) \leq N_P^2 N_s^2 \big(Adv^{\text{PKE}}(\mathcal{D}) + Adv^{\text{KDF}}(\mathcal{B})\big). \tag{6}$$

Case 2: The owner of the test session is not corrupted

In this case we consider the situation that $\mathcal{A}$ corrupts the partner of the test session but not the owner. The proof structure and games are similar to the previous case. The only difference in this case is that the algorithm $\mathcal{D}$ uses the public-key of the CCLA2 challenger as the public-key of the protocol principal U^* in Game 3′.

Game 1′:

This game is the original game. When the `Test` query is asked, the Game 1′ challenger chooses a random bit $b \leftarrow \{0,1\}$. If $b = 1$, the real session key is given to $\mathcal{A}$, otherwise a random value chosen from the same session key space is given.

Game 2′:

Same as Game 1′ with the following exception: before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \leftarrow \{U_1, \ldots, U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \leftarrow \{1, \ldots, N_s\}$ are chosen. The oracle $\Pi^{s^*}_{U^*,V^*}$ is chosen as the *target session* and the oracle $\Pi^{t^*}_{V^*,U^*}$ is chosen as the *partner* to the target session. If the target session is not the oracle $\Pi^{s^*}_{U^*,V^*}$ or the partner to the oracle $\Pi^{t^*}_{V^*,U^*}$, the Game 2′ challenger aborts the game.

Game 3′:

Same as Game 2′ with the following exception: the Game 3′ challenger chooses a random value $r' \leftarrow \{0,1\}^k$.

- If the test session is the initiator, the challenger computes $K_{U^*V^*} \leftarrow \text{KDF}(U^*, V^*, r_{U^*}, r')$.
- If the test session is the responder, the challenger computes $K_{U^*V^*} \leftarrow \text{KDF}(V^*, U^*, r', r_{U^*})$.

Game 4′:

Same as Game 3′ with the following exception: the Game 4′ challenger randomly chooses $K \leftarrow \{0,1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the `Test` query.

Differences between games

In this section the adversary's advantage of distinguishing each game from the previous game is investigated. $Succ_{\text{Game x}}(\mathcal{A})$ denotes the event that the adversary $\mathcal{A}$ wins Game x, $Adv_{\text{Game x}}(\mathcal{A})$ denotes the advantage of the adversary $\mathcal{A}$ of winning Game x.

Game 1′

is the original game. Hence,

$$Adv_{\text{Game 1}'}(\mathcal{A}) = Adv_{\pi}^{\text{CAFL}}(\mathcal{A}). \tag{7}$$

Game 1′ and Game 2′.

The probability of Game 2′ to be halted due to incorrect choice of the test session is $1 - \frac{1}{N_P^2 N_s^2}$. Unless the incorrect choice happens, Game 2′ is identical to Game 1′. Hence,

$$Adv_{\text{Game 2}'}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} Adv_{\text{Game 1}'}(\mathcal{A}). \tag{8}$$

Game 2′ and Game 3′.

We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$. If $\mathcal{A}$ can distinguish the difference between Game 2′ and Game 3′, then $\mathcal{D}$ can be used against the CCLA2 challenger of underlying public-key cryptosystem, PKE. The algorithm $\mathcal{D}$ uses the public-key of the CCLA2 challenger as the public-key of the protocol principal U^* and generates public/secret key pairs for all other protocol principals. $\mathcal{D}$ runs a copy of $\mathcal{A}$ and interacts with $\mathcal{A}$, such that it is interacting with either Game 2′ or Game 3′. $\mathcal{D}$ picks two random strings, $r'_0, r'_1 \leftarrow \{0,1\}^k$ and passes them to the CCLA2 challenger. From the CCLA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C such that $C \leftarrow \text{Enc}(pk_{U^*}, r')$ where $r' = r'_0$ or $r' = r'_1$. The procedure of answering each query is similar to the description in the "Game 2 and Game 3" in Case 1.

To compute the answer to the $\texttt{Test}(U^*, V^*, s^*)$ query, the algorithm $\mathcal{D}$ uses the r'_1 as the decryption of the ciphertext C and computes $\text{KDF}(U^*, V^*, r'_1, r_{V^*})$, if U^* is the initiator or computes $\text{KDF}(V^*, U^*, r_{V^*}, r'_1)$, if V^* is the initiator. If r'_1 is the decryption of C coming *to* the owner of the test session, U^*, the simulation constructed by $\mathcal{D}$ is identical to Game 2′ whereas if r'_0 is the decryption of C, the simulation constructed by $\mathcal{D}$ is identical to Game 3′. If $\mathcal{A}$ can distinguish the difference between Game 2′ and Game 3′, then $\mathcal{D}$ can distinguish whether $C \leftarrow \text{Enc}(pk_{U^*}, r'_0)$ or $C \leftarrow \text{Enc}(pk_{U^*}, r'_1)$.

The algorithm $\mathcal{D}$ plays the CCLA2 game against the public-key cryptosystem PKE according to the Definition 2.1 since $\mathcal{D}$ does not ask the decryption of the challenge ciphertext C. Hence,

$$|Adv_{\text{Game 2}'}(\mathcal{A}) - Adv_{\text{Game 3}'}(\mathcal{A})| \leq Adv^{\text{PKE}}(\mathcal{D}) \tag{9}$$

Game 3′ and Game 4′.

We introduce an algorithm $\mathcal{B}$ which is constructed using the adversary $\mathcal{A}$. If $\mathcal{A}$ can distinguish the difference between Game 3′ and Game 4′, then $\mathcal{B}$ can be used to distinguish whether the value K is computed using KDF or randomly chosen. $\mathcal{B}$ receives K from the KDF challenger, such that K is computed using the KDF or randomly chosen from the session key space. If K is computed using the KDF, the simulation constructed by $\mathcal{B}$ is identical to Game 3′ whereas if K is randomly

chosen, the simulation constructed by $\mathcal{B}$ is identical to Game $4'$. If $\mathcal{A}$ can distinguish the difference between Game $3'$ and Game $4'$, then $\mathcal{B}$ can distinguish whether the value K is computed using KDF or randomly chosen. Hence,

$$|Adv_{\text{Game } 3'}(\mathcal{A}) - Adv_{\text{Game } 4'}(\mathcal{A})| \leq Adv^{\text{KDF}}(\mathcal{B}) \tag{10}$$

Semantic security of the session key in Game $4'$.

Since the session key K of $\Pi^{s^*}_{U^*,V^*}$ is chosen randomly independently from all other values, $\mathcal{A}$ does not have any advantage in Game $4'$.

Hence,

$$Adv_{\text{Game } 4'}(\mathcal{A}) = 0 \tag{11}$$

Using equations (7)–(11) we find,

$$Adv^{\text{CAFL}}_{\pi}(\mathcal{A}) \leq N_P^2 N_s^2 \big(Adv^{\text{PKE}}(\mathcal{D}) + Adv^{\text{KDF}}(\mathcal{B})\big) \tag{12}$$

Combining Case 1 and Case 2

In both cases we can see the adversary $\mathcal{A}$'s advantage of winning against the protocol π challenger is

$$Adv^{\text{CAFL}}_{\pi}(\mathcal{A}) \leq N_P^2 N_s^2 \big(Adv^{\text{PKE}}(\mathcal{D}) + Adv^{\text{KDF}}(\mathcal{B})\big).$$

Hence, we conclude the proof of Theorem 4.1 saying that whenever the underlying public-key cryptosystem is CCLA2 secure and the key derivation function is secure with respect to the source of key material $\{0,1\}^{2k}$, the key exchange protocol π is CAFL-secure. □